[0062] <The second embodiment> Figs. 5 to 8 are diagrams for explaining a microprocessor 20 described in the second embodiment of the present invention. In the first embodiment, an encrypted program was operated using a public key inherent in microprocessors. However, an encryption/decryption algorithm of the public key method is generally more complicated than that of the common key method, requires a larger size circuit and a greater cost, and in addition, fast processing of the algorithm is difficult. Given these factors, the microprocessor in the second embodiment decrypts and operates an encrypted program using a common key and controls the common key safely and effectively.

[0063] To be specific, the operational program to be provided to the microprocessor is encrypted using any encryption key (that is, a common key) that the program vendor has selected. The common key used for encryption is encrypted using a public key inherent in microprocessors. Since both the encryption program encrypted by the common key and the common key encrypted by the public key are transmitted to the microprocessor, the microprocessor decrypts the common key with a secret key inherent in the microprocessor, and the encryption program is decrypted with the decrypted common key. By doing this, hardware costs for decryption may be reduced while maintaining security.

[0064] As shown in Fig. 5, the chipped or packaged microprocessor 20 includes a secret key 212 that cannot be read from the program; a public key 214 corresponding to the secret key; a decryption key TLB (a first decryption means) 236 that decrypts a third encryption key Ekp [Kx] stored in an external memory in a encryption state beforehand with the secret key 212; a decryption processing unit 216 (the second decryption processing means) that decrypts a encrypted program stored in the external memory using the decrypted third encryption key Ekp [Kx]; a primary instruction cache 218 that stores decrypted programs; a bus interface unit 222; and an instruction TLB 220.

[0065] The bus interface unit 222 has an address conversion means (not shown in the figures) that converts a virtual address into a physical address and a memory read means (not shown in figures) that reads the content corresponding to the designated physical address. The instruction TLB 220 loads the table that includes the conversion rule for converting from a virtual address into a physical address used in the address conversion means and the key information for the third decryption key E [Kx] decrypted in an external memory using the public key. To be more specific, the instruction TLB 220 loads a page table including entries corresponding to each page in a page area separated

by the address of the main memory 203. Each entry has an encryption flag that shows the encryption attribute of the corresponding page and an identifier (key entry ID) that designates the position on the key table that stores key information for decrypting the contents of the page. By referring to the identifier, the corresponding key entry is loaded.

[0066] Fig. 6 shows a file format that stores encryption programs operated in the microprocessor in accordance with the second embodiment. In the field of the text section header 63, fields of the encryption algorithm 63-2 and an encryption key value 63-3 are added. In the field of the encryption algorithm 63-2, an encryption algorithm for encrypting a code area is coded and stored. In the field of an encryption key value 63-3, the value Ekp[kx] encrypted from a common key Kx by the public key 214 is stored, where the common key Kx is used by the microprocessor 20 to read an encrypted program using the public key 214 inherent in the microprocessor. The length required for the common key Kx is, for example, 192 bits when a triple DES is used in encryption. In addition, the size of the Ekp[Kx] becomes several times the length of a block when it is encrypted by the public key 214 in the microprocessor 20.

[0067] The '.text' section 66 is encrypted by the encryption algorithm designated in the field of the encryption algorithm 63-2 using the common key Kx. Here, a triple DES in the common key method is designated to the encryption algorithm. In the characteristic field 63-1, it is the same as the first embodiment in that the encryption flag is set to 1 to indicate that the program is being encrypted.

[0068] For a program vendor to make a encryption file shown in Fig. 2, first, randomly select an encryption key in a code area as a common key Kx and encrypt the text section 66 by a block unit of 64 bits (8 bites) using the triple DES algorithm. In the encryption key 63-3, the value Ekp [Kx] that encrypted from the common key Kx for reading using the targeted public key 214 in the microprocessor 20 is stored and in the encryption algorithm field 63-2, a code expression of the triple DES is included and the encryption flag in the characteristic field is set to 1.

[0069] It should be noted that by making the area that stores codes to be plural sections, not limited to text section 66, the code of the same program can be encrypted using a plurality of encryption keys or encryption algorithms. However, each section should be positioned by the page boundaries in the virtual memory (Fig. 7) of the processor.

[0070] In addition, it is possible to provide a part that does not relate to the program operation in a portion of the code area in the program, and to directly store the public key 214 in the targeted microprocessor 20 that uses the program in this portion and a program-distributed user ID before encrypting the code area, or to store them in the

form of encryption. By doing this, it becomes possible to track the route when an unauthorized decoding of a program is performed and unauthorized copies are distributed. Since information for tracking the route is coded in a sequence of a machine language instruction that can be operated as one sub routine of a program, it is possible to make it difficult to find.

[0071] Now, when operating an encrypted program in the microprocessor 20, as in the first embodiment, first, it begins with loading an encryption program using a loader to the main memory 203 in the PC having the microprocessor 20. The loader sets a process context including a virtual memory space for the encryption program and positions all sections on the memory in the same way and performs relocations etc. according to necessity.

[0072] Fig. 7 shows a page table when operating an encryption program in the microprocessor in the second embodiment. The page table in the second embodiment shown in Fig. 7 is an extension table having a key table 79. A logical address 71 is divided into the three fields of a directory 72, a table 73, and an offset 74. The three fields that comprise the logical address 71 calculate a physical address as follows.

[0073] First, the top directory field designates the entry 76-i of the directory table 76. The forefront of the directory table 76 is designated by the register 75 in the microprocessor 20. The directory entry 76-i has a pointer at the forefront of the page table 77. The position of the page table 77 is calculated by the table 73 in the logical address 71, and in the corresponding page entry 77-i, the actual physical address 80-k shown by the address is described. Thus far, it is the same as the first embodiment.

[0074]Fig. 8 shows the entry of the page table 77 and the corresponding entry of the key table 79 of Fig. 7. In the second embodiment, a field 77-j-k that stores the number (identifier or key entry ID) that designates the key information for decrypting the corresponding page is provided to each entry of the page table 77. This number designates the entry number in which the decryption key is stored. The forefront of the key table 79 is designated by the key table control register 78, one of the control registers in the microprocessor 20. In addition, same as in the first embodiment, each entry has a decryption flag 77-j-E and a debug flag 77-j-D that indicate attributes of the corresponding page.

[0075] Next, the loader secures the entry 79-m for storing the value Ekp [Kx] that decrypted the common key Kx for decrypting decrypted programs by the public key 214 inherent in the microprocessor 20 on the key table 79. In the key table 79, there is a field 79-m-1 of the reference counter in addition to the fields of the key information and the decryption algorithm, and the number of tables referring to the table is written.

That is, if the reference number is 0, the table can be used. The loader determines the table to be used and after rewriting the reference counter in the key table from 0 to 1, the key information and the encryption algorithm information is written in the field 79-m-3 and the 79-m4 in the key table, respectively and the number of the key entry is written in the entry 77-j in the corresponding page table 77. If there are a plurality of pages that refer the same key entry, the number of the key entry is written in the page table 77 after increasing the number of reference counters in the key table 79.

[0076] If the same key is stored in the key table 79 because the same program is being operated or some other similar reason, the securing of tables is not performed and the number of the key entry is written in the key entry after increasing the number of reference counters in the key table. Thus, the encryption key E [Kx] can be referred to by a plurality of pages.

[0077] As described above, the reason for enabling common decryption key information to be shared between plural pages by providing an independent key table 79 is to improve the efficiency of the instruction TLB 220 by decreasing the amount of memory required for the page table. That is, since the same public key 114 was used for all the programs in the encryption programs in the first embodiment, storing key information for each page table was not required; however, in the second embodiment, the key information of the common information required for each program is different and storing different keys for each page is necessary. However, if key information that requires a relatively long bit length is directly allocated to the field of a page table, the size of the page table increases and the use efficiency of the memory 203 is deteriorated. For example, if a 32-byte key whose size is the same as a cache line is allocated to each page of the page table when a page is 4K-byte, the result is that memory a little less than 1% of the page size is required for the page table. Even if the size of the page table is enlarged, if the size of the instruction TLB 220 is the same as before, the number of hits of the instruction TLB 220 for instruction reference becomes decreased and the operation speed of the program becomes deteriorated. It is possible to enlarge the size of the instruction TLB 220 at the same time as well; however, since the instruction TLB 220 has a character that requires high-speed conversion of a logical address and a physical address, a higher cost results.

[0078] Moreover, in fact, the types of encryption keys are not necessarily changed for each page and often, the use of several types of keys at most is enough to protect the confidentiality of the program. Given this factor, in the second embodiment, an independent key table 79 is provided so that the same decryption key information is shared among plural pages, and by storing the number to the key table 79 from the key

table 77, the amount of memory required for the page table 77 is deteriorated.

[0079] Now, when positioning of programs and setting of key information of the page table are finished, the loader sets the encryption flag of the page table in which the '.text' section header 63 is positioned to 1. If the encryption flag of the page table is set to 1 when the page table is cached in the instruction TLB 220, the bus interface unit 222 monitors it and makes corresponding entries of the primary instruction cache 218 and the second instruction cache 205 invalid.

[0080] The value of the key table 79 described above is stored in the decryption key TLB 26 as well. When the key table 79 is rewritten, the entry of the corresponding decryption key TLB 236 is also rewritten. Here, the decryption key information E [Kx] in the memory is decrypted by the public key 214 inherent in the microprocessor 20 and stored in the decryption key TLB 220. Then, by making the length of the decryption key information E [Kx] on the memory 203 longer than that of the decrypted key information Kx on the decryption key TLB 220, the safety of the key and the use efficiency of the TLB 220 can be balanced. This is because the key information stored in the decryption key TLB 220 placed inside the microprocessor 20 cannot be seen from the user program. Needless to say, for the explicit memory reference of the key table by a user program, not the decrypted value Kx but the former encrypted value E[Kx] is returned.

[0081] When setting of the key table 79 is completed, as in the first embodiment, the loader transfers control to the entry point of the program and the operation of the program is started. When the processor accesses to the encryption area as an instruction, a cache miss-hit is occurred and reading from the main memory 203 to the first instruction cache 218 is started. Then, the entry of the key table 79 designated by the reference from the page table 77 to the key table 79 is decrypted by the secret key 212 in the microprocessor and is read to the decryption TLB 236. The encryption data stored in the main memory 203 using the decrypted key information Kx is decrypted by the decryption processing unit 216 and is read to the primary instruction cache 218 and executed.

[0082] The invalidation of the cache and interrupt process following the completion of operations may be performed thereafter as the same in the first embodiment.

[0083] As described above, in the second embodiment, the common key method can be applied for the decryption of a program. A hardware implementation of the common key algorithm is generally easier than that of a public key algorithm with the same key length and it is possible to reduce the cost of the decryption function. Moreover, since the common key is encrypted using a public key inherent in the microprocessor 20 and

the decryption cannot be done without an inherent secret key in the microprocessor 20, such algorithm will be secured.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2001－230770
（P2001－230770A）
(43)公開日　平成13年８月24日(2001.8.24)

(51)Int.Cl.⁷	識別記号	FI	テーマコード（参考）
H04L 9/10		G09C 1/00	620Z 5J104
G09C 1/00	620	H04L 9/00	621A 9A001

審査請求　未請求　請求項の数20　ＯＬ　（全 22 頁）

(21)出願番号　特願2000－35898(P2000－35898)

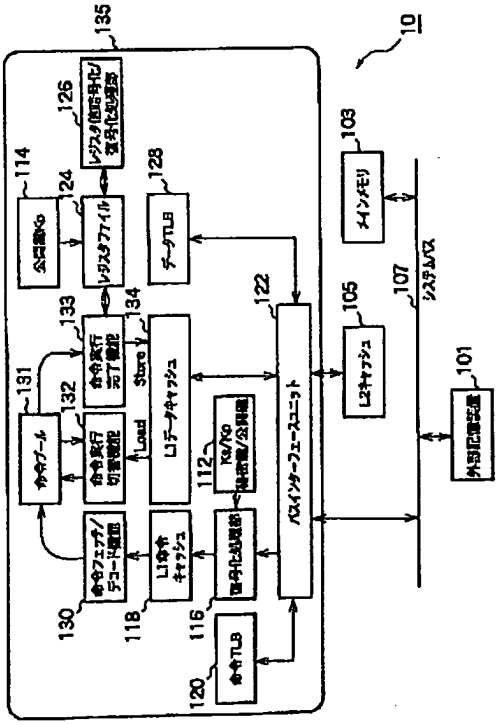
(22)出願日　平成12年２月14日(2000.2.14)

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　橋本　幹生
神奈川県川崎市幸区小向東芝町１　株式会社東芝研究開発センター内
(72)発明者　斉藤　健
神奈川県川崎市幸区小向東芝町１　株式会社東芝研究開発センター内
(74)代理人　100083806
弁理士　三好　秀和　（外７名）

最終頁に続く

(54)【発明の名称】　マイクロプロセッサ

(57)【要約】
【課題】　実行プログラムの不正な書き換えを確実に防止することのできるマイクロプロセッサを提供する。
【解決手段】　マイクロプロセッサは、外部へ読み出し不可能な固有の秘密鍵を内部に保持し、この秘密鍵に対応する固有の公開鍵であらかじめ暗号化された内容を復号化する。マイクロプロセッサのバスインターフェイスユニットは、仮想アドレスを物理アドレスに変換するアドレス変換手段と、マイクロプロセッサ外部のメモリから指定された物理アドレスに対応する内容を読み出すメモリ読み出し手段とを含む。命令ＴＬＢは、アドレス変換規則と、各々が仮想アドレスで指定された範囲の暗号化属性情報を有する１以上のエントリとを含むテーブルを格納する。復号化処理部は、暗号化属性情報が、仮想アドレスで指定された範囲が暗号化されていることを示す場合に、対応する内容をメモリ読み出し手段を介して外部メモリから読み出し、復号化する。



するプログラム配布装置であって、
前記クライアント装置との間に第１の通信路を設定する第１通信路設定部と、
前記第１の通信路を介して前記クライアント装置を使用するユーザの認証を行うユーザ認証部と、
前記第１の通信路上に、クライアント装置が内蔵するマイクロプロセッサに直接連絡する第２の通信路をさらに設定する第２通信路設定部と、
前記クライアント装置に配布すべき実行プログラムを暗号化して暗号化プログラムを作成する暗号化処理部と、
前記暗号化したプログラムを前記第２の通信路を介してクライアント装置のマイクロプロセッサに送信する配布送信部とを備えたプログラム配布装置。
【請求項１２】　前記第２の通信路を介して、クライアント装置のマイクロプロセッサが正当であることを認証するプロセッサ認証部をさらに有し、前記プロセッサ認証部は、前記第２の通信路を介して、クライアント装置のマイクロプロセッサが、該マイクロプロセッサに固有の公開鍵と秘密鍵とを確かに保持することを認証することを特徴とする請求項１１に記載のプログラム配布装置。
【請求項１３】　前記暗号化処理部は、前記第２の通信路を介して前記クライアント装置のマイクロプロセッサから固有の公開鍵を受け取り、該公開鍵を用いて前記実行プログラムを暗号化することを特徴とする請求項１１または１２に記載のプログラム配布装置。
【請求項１４】　前記暗号化処理部は、任意の暗号化鍵で前記実行プログラムを暗号化し、前記第２の通信路を介してクライアント装置のマイクロプロセッサから固有の公開鍵を受け取り、前記任意の暗号化鍵を前記公開鍵で暗号化し、
前記配布送信部は、前記暗号化した実行プログラムと、暗号化された暗号化鍵とを、前記第２の通信路を介してクライアント装置のマイクロプロセッサに送信することを特徴とする請求項１１または１２に記載のプログラム配布装置。
【請求項１５】　ネットワークを介してプログラム配布装置からプログラムの配布を受けるクライアント装置であって、
あらかじめ固有の秘密鍵と公開鍵とを有するマイクロプロセッサと、
前記プログラム配布装置との間に第１の通信路を設定する第１クライアント側通信路設定部と、
前記第１の通信路を介して、前記クライアント装置を使用するユーザのユーザＩＤを前記プログラム配布装置に送信するユーザ認証部と、
前記第１の通信路上に、前記マイクロプロセッサから直接プログラム配布装置に連絡する第２の通信路をさらに設定する第２クライアント側通信路設定部と、
前記第２の通信路を介してプログラム配布装置に、前記マイクロプロセッサが該マイクロプロセッサに固有の秘密鍵と公開鍵とを確かに保持することを証明する証明を送信する証明部と、
前記第２の通信路を介して前記プログラム配布装置から、暗号化された実行プログラムを受信する受信部とを有することを特徴とするクライアント装置。
【請求項１６】　ネットワークと、
前記ネットワークに接続され、実行プログラムを前記ネットワークを介して配布するプログラム配布装置と、
前記ネットワークに接続され、ネットワークを介して前記プログラム配布装置から実行プログラムの配布を受けるクライアント装置とを含み、
前記クライアント装置は、
あらかじめ固有の秘密鍵と公開鍵とを有するマイクロプロセッサと、
前記プログラム配布装置との間に第１の通信路を設定する第１クライアント側通信路設定部と、
前記第１の通信路上に、前記マイクロプロセッサから直接プログラム配布装置に連絡する第２の通信路をさらに設定する第２クライアント側通信路設定部と、前記第２の通信路を介してプログラム配布装置に、公開鍵および該公開鍵が確か前記マイクロプロセッサに固有のものであることを証明する証明を送信する証明部と、前記第２の通信路を介して前記プログラム配布装置から、実行プログラムを受信する受信部とを備え、
前記プログラム配置装置は、
前記クライアント装置との間に第１の通信路を設定する第１通信路設定部と、
前記第１の通信路を介して前記クライアント装置を使用するユーザの認証を行うユーザ認証部と、
前記第１の通信路上に、クライアント装置が内蔵するマイクロプロセッサに直接連絡する第２の通信路をさらに設定する第２通信路設定部と、
前記クライアント装置に配布すべき実行プログラムを暗号化して暗号化プログラムを作成する暗号化処理部と、
前記暗号化したプログラムを前記第２の通信路を介してクライアント装置のマイクロプロセッサに直接送信する配布送信部とを備えることを特徴とするプログラム配布システム。
【請求項１７】　固有の秘密鍵と、この秘密鍵に対応する固有の公開鍵とをあらかじめ有するマイクロプロセッサを内蔵するコンピュータから、プログラム配布装置に、ネットワークを介して第１の通信路を設定するステップと、　前記第１の通信路上に、前記マイクロプロセッサからプログラム配布装置に直接連絡する第２の通信路をさらに設定するステップと、
前記第２の通信路を介して、前記マイクロプロセッサから前記プログラム配布装置に、前記固有の公開鍵を送信するステップと、
前記ネットワークを介して送信された公開鍵を、プログ

るだけではなく、ＰＣソフトウエア自体に含まれるノウハウなどを解析から守るのにも有効である。

【０００６】

【発明が解決しようとする課題】しかしながら、耐タンパソフトウエア技術の根本原理は、プログラムの中で保護を要する部分を実行前は暗号化しておき、実行する直前に復号し、実行終了時に再び暗号化することにより、逆アセンブラ、デバッガなどの解析ツールによる解析を困難にするものである。したがって、プログラムがプロセッサによって実行可能である以上、プログラム開始時から順を追って解析していけば必ず解析することが可能である。換言すれば、耐タンパソフトウエア技術を用いても、プログラムコードを逐次的に解析することによって、プログラムの動作が解読可能となる。

【０００７】この事実は、ＰＣを利用して動画像や音声を再生するシステムに、著作権者が著作物を供給する際の妨げとなる。また、ＰＣを通じて高度な情報サービスや企業、個人のノウハウを含んだプログラムをＰＣに適用しようにも、不正コピーの可能性が依然存在する以上、適用が見合わせられ、ＰＣソフトウエアの応用範囲が著しく狭められるという問題もある。

【０００８】そこで本発明の第１の目的は、実行プログラムの書き換えを確実に防止することができ、ＰＣソフトウエアの広範囲な適用を可能にするマイクロプロセッサの提供にある。

【０００９】本発明の第２の目的は、必要な暗号化回路、復号化回路、メモリのコストを削減し、安価かつ高性能なマイクロプロセッサの提供にある。

【００１０】本発明の第３の目的は、ネットワークを介して、クライアント側装置に安全にプログラムを配布することのできるプログラム配布装置の提供にある。

【００１１】本発明の第４の目的は、ネットワークを介して配布されたプログラムを安全に受信することのできるクライアント側装置の提供にある。

【００１２】本発明の第５の目的は、ネットワークを介して安全にプログラムを配布することのできるプログラム配布システムの提供にある。

【００１３】本発明の第６の目的は、公衆ネットワークを介した安全なプログラムの配布方法の提供にある。

【００１４】本発明の第７の目的は、暗号化されたプログラムを、オペレーティングシステムとの不整合なくロードすることのできるプログラムロード方法の提供にある。

【００１５】本発明の第８の目的は、第三者による不正な解析から保護することを可能にした、実行ファイルを記録したコンピュータ読み取り可能な記録媒体の提供にある。

【００１６】

【課題を解決するための手段】上記の第１の目的を達成するために、本発明のマイクロプロセッサは、外部へ読み出すことのできない固有の秘密鍵を内部に保持し、この秘密鍵に対応する公開鍵であらかじめ暗号化された内容を復号化する１チップまたは１パッケージのマイクロプロセッサを提供する。このマイクロプロセッサは、仮想アドレスを物理アドレスに変換するアドレス変換手段と、このアドレス変換手段の変換規則と各々が前記仮想アドレスで指定された範囲の暗号化属性情報を有する１以上のエントリとを含むテーブルを格納する第１の記憶手段（たとえば変換索引バッファ）と、マイクロプロセッサ外部のメモリから指定された物理アドレスに対応する内容を読み出すメモリ読み出し手段と、前記テーブルのエントリに含まれる暗号化属性情報が、仮想アドレスで指定された範囲が暗号化されていることを示す場合に、外部メモリに記録された内容をメモリ読み出し手段を介して読み出し、前記固有の秘密鍵によって復号化する復号化手段と、復号化手段で復号された内容を一時的に記憶する第２の記憶手段と（たとえば一次キャッシュ）、第２の記憶手段に記憶された内容を逐次解釈する命令デコード手段とを備える。アドレス変換手段と、メモリ読み出し手段は、たとえばバスインターフェイスとして実現され得る。

【００１７】復号化され、第２の記憶手段に格納された内容は、命令デコード手段以外からは読み出し禁止とされる。また、仮想アドレスで使用している外部メモリの対応する物理アドレスに対して読み出しが行われた場合には、この読み出しに対して例外処理を発生させるか、または物理アドレスに記録されている値とは異なる値を返す。このように、暗号化された内容の復号化処理は、マイクロプロセッサに固有の秘密鍵によって、完全にマイクロプロセッサ内部で行われ、かつ復号化されたデータの外部からの読み出しが防止される。すなわち、ＰＣの所有者によるプログラムの解析、改変が効果的に防止される。また、暗号化属性を示す情報、たとえば暗号化フラグをエントリに設定することにより、既存のプログラムにソース変更や再コンパイルなどの変更を加えることなく、暗号化フラグの状態を見ることによって、暗号化されたプログラムを滞りなく実行することが可能になる。

【００１８】また、暗号化プログラムの実行中の割り込みの発生を考慮して、マイクロプロセッサは、復号化された内容の実行状態を表わす値を保持するレジスタと、暗号化された内容の実行中に割り込みが発生した場合に、レジスタに記憶されている値を、ランダムな値を持つ任意の暗号化鍵で暗号化するレジスタ暗号化／復号化処理部とをさらに有する。暗号化に用いた任意の暗号化鍵自体を、マイクロプロセッサに固有の秘密鍵でさらに暗号化して、レジスタに保存する。割り込みが終了して、暗号化プログラムを続行する場合は、レジスタ値暗号化／復号化処理部は、レジスタの内容を復号化する。この構成により、不正な解析をより困難にし、安全性を

暗号化してある。プログラムは、コンピュータを介さずに、直接マイクロプロセッサに送信され、この公開鍵と対応する固有の秘密鍵を有するマイクロプロセッサでなければ復号することができない。このような配布方法により、第三者による不正な復号を効果的に防止することができる。

【００２８】第７の目的を達成するために、本発明のプログラムロード方法は、暗号化されたプログラムに対して１以上のページからなる仮想記憶領域を割り当て、前記暗号化されたプログラムを前記１以上のページに書き込んでから、前記１以上のページのそれぞれに対応するページテーブルエントリの暗号化フラグをセットする。これにより、既存のオペレーティングシステムとの不整合を生じさせることなく、暗号化されたプログラムを実装することができる。また、このようなロード方法を実行させるプログラムをあらかじめコンピュータ読み取り可能な記録媒体に記録してもよい。ここでいう記録媒体とは、たとえばフロッピーディスク、ＣＤ－ＲＯＭ、ＭＯディスクなどのコンピュータ外部のメモリ装置、半導体メモリ、磁気ディスク、光ディスク、磁気テープなどを含む。

【００２９】さらに、第８の目的を達成するために、暗号化されたプログラムファイルを記録するコンピュータ読み取り可能な記録媒体を提供する。このプログラムファイルは、外部のプログラムへの直接の参照を含まない１以上の第１のプログラム領域と、外部のプログラムへの直接の参照を含む１以上の第２のプログラム領域を有する。第１プログラム領域は、あらかじめ定められた暗号化鍵で暗号化された実行ファイルを記録し、第２プログラム領域は、外部のプログラムモジュールへのジャンプ命令を並べたアドレス変換用のテーブルを平文の状態で記録する。このプログラムファイルがプログラムローダによりコンピュータのメモリ上に読み込まれるとき、外部への参照を含む第２領域は、そのコンピュータの外部プログラムの配置に基づいて適切なアドレスを参照するようにプログラムローダによって書き換えられる。このとき、第２領域が平文の状態でアドレス変換テーブルを格納しているので、暗号化された実行プログラムをロードするときにリロケーションで不整合が生じない。また、ロード時の不整合を防止すると同時に、暗号化されてデータを格納する第１領域により、プログラムの大部分を不正な解析から保護することが可能になる。本発明のその他の特徴、効果は、以下に述べる実施の形態によって、より明確になるものである。

【００３０】

【発明の実施の形態】以下、図面を参照して本発明を詳細に説明する。

【００３１】＜第１実施形態＞図１～図４は、本発明の第１実施形態に係るマイクロプロセッサ１０を説明するための図である。図１の概略ブロック図に示すように、マイクロプロセッサ１０は、このマイクロプロセッサに固有の秘密鍵１１２と、この秘密鍵とペアをなす公開鍵１１４と、この公開鍵によりあらかじめ暗号化された実行プログラムを実行する際に秘密鍵１１２で復号化する復号化処理部１１６と、復号化されたプログラムを一時的に格納する一次命令キャッシュ１１８と、一次命令キャッシュ１１８に格納された内容を逐次解釈する命令フェッチ／デコード手段２３０と、バスインターフェイスユニット１２２と、命令ＴＬＢ１２０とを、１チップ上または１パッケージ１３５内に備える。

【００３２】バスインターフェイスユニット１２２は、仮想アドレスを物理アドレスに変換するアドレス変換手段（不図示）と、マイクロプロセッサ外部のメインメモリ２０３から、指定された物理アドレスの内容を読み出すメモリ読み出し手段（不図示）とを有する。また、命令ＴＬＢ１２０は、アドレス変換手段の変換規則と、各々が仮想アドレスで指定された範囲が暗号化されているかどうかを示す暗号化フラグを有する１以上のエントリとを含むページテーブルを読み込む。

【００３３】マイクロプロセッサ１０は、たとえばＰＣに組み込まれると、システムバス１０７を介して外部記憶装置１０１、ＰＣのメインメモリ１０３、２次キャッシュ１０５に接続される。ユーザは、所望のプログラムを暗号化された状態で購入したい場合に、このマイクロプロセッサ１０に固有の公開鍵を、ＰＣを介してプログラムベンダに送信する。プログラムベンダは、送られてきた公開鍵でプログラムを暗号化して、暗号化された実行プログラムをユーザのＰＣに送り返す。暗号化された実行プログラムは、たとえばＰＣのメインメモリ１０３に図２に示すファイル形式でロードされ、実行する場合にだけマイクロプロセッサ１０の復号化処理部１１６で復号化され、実行される。このとき、マイクロプロセッサ１０に固有の秘密鍵１１２を用いなければプログラムの復号化はできない。

【００３４】このマイクロプロセッサ１０は仮想記憶性能を有する。したがって、図３に示すように、メインメモリ１０３の所定の範囲を指定する物理アドレスによって区切られたページ３８－１～３８－ｋと、これらに対応するエントリを有するページテーブル３７が設定される。命令ＴＬＢ１２０は、マイクロプロセッサが実行しようとする範囲のプログラムが書き込まれた物理ページに対応するページテーブルエントリを取り込む。各エントリには、上述したように暗号化フラグが設けられており、対応するページの内容が暗号化されている場合に暗号化フラグがセットされる。復号化処理部１１６は、エントリに暗号化フラグがセットされている場合にのみ、このエントリに対応するページの内容を復号化する。ページテーブルと暗号化フラグの詳細については、後述する。

【００３５】暗号化プログラムの実行中は、復号化され

指定している。

【００４４】第１実施形態では、特性フィールド１３－１上の１ビット、たとえば０ｘ００００４００でマスクされる１ビットが、暗号化属性を指定する暗号化フラグとして用いられる。このビットが１であれば、対応のセクションが暗号化されていることを示し、０であれば暗号化されていないことを示す。図２の例では、このビットが値１を有するので、対応のセクションは暗号化されていることになる。たとえば、.textセクション１７の内容は、本来平文であったプログラムをマイクロプロセッサ１０の固有の公開鍵１１４で暗号化されたものである。一方、ジャンプテーブルが格納される.IATセクションヘッダ１６の暗号化フラグは値０を持ち、このセクションにはジャンプテーブルが暗号化されず、平文のまま格納されることを示す。

【００４５】一般に、プログラムがロードされるアドレスは、システム構成や、そのメモリの利用状況によって異なる。そのようなとき、プログラムに含まれる参照アドレスをプログラムローダが状況に合わせて変更する必要が生じる。これをリロケーションと呼ぶ。あるプログラム内部に閉じた関数呼出しは、プログラム内部の相対的なアドレスに基づいてアクセスを行うリロケータブルなコードを生成することによって、リロケーション処理を不要にすることができる。しかし、プログラム外部のモジュールの呼出し、たとえばシステムコール呼出しや、その逆に外部のモジュールからプログラム内部の関数が呼び出される場合には、モジュールの絶対アドレスも相対的な位置関係も、プログラムがロードされるまでわからないことが多い。システムのハードウエア構成やソフトウエアのバージョンによってプログラムや作業領域のサイズが変わることがその理由である。

【００４６】この問題を解決するため、次のような手法を用いる。外部関数への呼出しを含むプログラムファイルに、ＩＡＴ（Import Address Table）と呼ばれる、読み出される外部関数へのジャンプ命令を並べたテーブルをあらかじめ作成しておき、暗号化されたプログラム本体（ここでは.textセクション１７）から外部モジュールへの呼出しは、このテーブルへのコール命令として間接的に行う。ジャンプテーブルには、呼出し先の関数を識別する名前が付加されており、プログラムローダはプログラムのロード時に名前に基づいて外部関数のアドレスを検索し、上記テーブルの対応する命令のジャンプ先をその関数のアドレスに書き換える。この状態でプログラムから外部関数への呼出しが行われると、制御は一度ＩＡＴのジャンプ命令に移り、次にジャンプ命令の飛び先の目的とする関数に制御が移る。第１実施形態では、.IATセクション２０にこのジャンプテーブルが格納されている。

【００４７】ＩＡＴ領域はプログラムローダによる書き換えが行われるので、この領域が暗号化されていると、書き換えが正常に行われなくなり、正しい外部関数が呼び出されないことになる。そこで本発明では、図２に示すように、ＩＡＴをプログラム本体（.textセクション１７）とは別のセクションに設け、.IATセクション２０を平文のまま維持することにより、リロケーション処理が正しく行われる。ＩＡＴに含まれる情報は、外部関数への呼出しだけなので、プログラム自体の秘密が損なわれることはない。

【００４８】第１実施形態では、ＩＡＴを外部関数へのジャンプ命令が格納されていることとしたが、プログラム本体からの呼出し命令が、ＩＡＴの内容で指定されるような間接呼出し命令を使っている場合には、ＩＡＴの内容はジャンプ命令ではなく、単なるとび先のアドレスが格納されたデータ列であってもよい。もちろんこの場合も、ＩＡＴは暗号化されない。

【００４９】次に、ページテーブルエントリに設けられた暗号化フラグについて説明する。暗号化された実行プログラムは、ローダプログラムによってメインメモリ１０３上に読み込まれる。このとき、ローダは、ロードするプログラムのために１以上のページから成る仮想メモリ空間を割り当て、これらのページにプログラムのすべてのセクションを書き込んで配置する。配置が終わると、ローダは.textセクション１７が配置されたページテーブルエントリ37-jの暗号化フラグ37-j-E（図４）を１にセットする。このページテーブルエントリ37-jが、マイクロプロセッサ１０の命令ＴＬＢ１２０にキャッシュされているときに暗号化フラグがセットされていると、バスインターフェイスユニット１２１がそれを監視していて、一次命令キャッシュ１１８と二次命令キャッシュ１０５の、このページエントリに対応するキャッシュエントリを無効化して、復号化された内容を記憶するためのスペースをあける。このとき、平文状態の.IATセクションが配置されたページテーブルエントリの暗号化フラグは０のままである。

【００５０】いったん、ページテーブル307-j-Eの暗号化フラグがセットされると、以後このページテーブルに対応する物理メモリのページ38-kへの他プログラムからのアクセスは禁止される。すなわち、このページへの読み出し、書き込みは一切できなくなる。この保護は、実行がユーザモード、カーネルモードのどちらの場合にも適用される。

【００５１】暗号化フラグがセットされている間に、外部から読み書きの要求があった場合は、保護例外が発生する。あるいは、読み出しに対しては、常に所定の値（たとえば０）もしくは乱数が読み出され、書き込みに対しては無効とするようにしてもよい。例外の扱いについては後述する。暗号化フラグがセットされた物理ページ310-k が平文状態で一次データキャッシュ１３４にキャッシュされることはない。以下、ページテーブルの暗号化フラグがセットされたページに対応するメモリ領域

のページに対して暗号化プログラムの実行以外の読み書きが行われると、暗号化プログラムの保護例外が発生する。しかし、デバッグ作業は平文状態で行われているので、実行コンテキストは平文のままセーブされ、ステップ実行も可能となる。すなわち、プログラム実行においては、デバッグは平文のプログラムであっても、振る舞いは暗号化された状態と一致する。かつ、ステップ実行による実行トレースが可能になる。また、保護による例外が発生した場合も、セーブされた実行コンテキストを解析することにより、例外の発生原因を調べた上で、実行を再開することもできる。

【００６０】ページテーブルについては、暗号化プログラムと平文のプログラムの間で暗号化フラグの値の相違は残るが、一般にアプリケーションプログラムはページテーブルの内容を意識することはないので、アプリケーションプログラムのデバッグの妨げにはならない。

【００６１】暗号化プログラムのセキュリティを守るため、暗号化フラグがセットされた状態では、デバッグフラグをセットしても、ステップ実行などは禁止されたままにする。または、暗号化フラグがセットされた状態では、デバッグフラグをセットできないようにしてもよい。なお、暗号化フラグがセットされている限り、デバッグフラグの状態に関わらず、暗号化領域からのデータ読み出しは禁止される。

【００６２】＜第２実施形態＞図５～図８は、本発明の第２実施形態に係るマイクロプロセッサ２０を説明するための図である。第１実施形態では、マイクロプロセッサに固有の公開鍵を用いて暗号化されたプログラムを実行していた。しかし、公開鍵方式の暗号化／福号化アルゴリズムは、共通鍵方式に比較して一般に複雑で、回路規模が大きくなり、コストが高くなる上に、処理の高速化が困難である。そこで、第２実施形態のマイクロプロセッサは、共通鍵で暗号化されたプログラムを復号し実行するとともに、共通鍵を安全かつ効率的に管理する。

【００６３】具体的には、マイクロプロセッサに供給される実行プログラムは、プログラムベンダが選択した任意の暗号化鍵（すなわち共通鍵）で暗号化される。暗号化に用いた共通鍵は、マイクロプロセッサに固有の公開鍵で暗号化される。共通鍵で暗号化された暗号化プログラムと、公開鍵によって暗号化された共通鍵とがマイクロプロセッサに送られてくるので、マイクロプロセッサは、固有の秘密鍵で共通鍵を復号化し、復号化された共通鍵で、暗号化プログラムの内容を復号化する。これにより、セキュリティを維持したまま、復号化のためのハードウエアコストを低減することができる。

【００６４】図５に示すように、マイクロプロセッサ２０は、プログラムから読み出すことのできない秘密鍵と２１２、この秘密鍵に対応する公開鍵２１４と、この公開鍵であらかじめ暗号化された状態で外部のメモリに格納されている第３の暗号化鍵ＥＫｐ［Ｋｘ］を前記秘密鍵２１２を使用して復号化する復号化鍵ＴＬＢ（第１の復号化処理手段）２３６と、外部のメモリに格納されている暗号化されたプログラムを、前記復号化された第３の暗号化鍵ＥＫｐ［Ｋｘ］によって復号化する復号化処理部２１６（第２の復号化処理手段）と、復号化されたプログラムを格納する一次命令キャッシュ２１８と、バスインターフェイスユニット２２２と、命令ＴＬＢ２２０とを、１つのチップまたはパッケージ内に備える。

【００６５】バスインターフェイスユニット２２２は、仮想アドレスを物理アドレスに変換するアドレス変換手段（不図示）と、マイクロプロセッサ外部のメインメモリ２０３から、指定された物理アドレスに対応する内容を読み出すメモリ読み出し手段（不図示）とを有する。命令ＴＬＢ２２０は、アドレス変換手段による仮想アドレスから物理アドレスへの変換規則と、公開鍵によってあらかじめ外部で暗号化された第３の暗号化鍵Ｅ［Ｋｘ］（共通鍵）についてのキー情報とを含むテーブルを取り込む。より具体的には、命令ＴＬＢ２２０は、まずメインメモリ２０３のアドレスによって区切られたページ領域の各ページに対応するエントリを含むページテーブルを取り込む。各エントリは、対応するページの暗号化属性を示す暗号化フラグと、このページの内容を復号するための鍵情報が記憶されているキーテーブル上の位置を指定する識別子（キーエントリＩＤ）とを有する。この識別子を参照することによって、対応するキーテーブルのエントリを取り込む。

【００６６】図６は、第２実施形態のマイクロプロセッサ２０で実行する暗号化プログラムを格納するファイル形式を示す。.textセクションヘッダ６３のフィールドには、暗号化アルゴリズム６３－２のフィールドと、暗号化鍵値６３－３のフィールドが追加されている。暗号アルゴリズム６３－２のフィールドには、コード領域を暗号化するための暗号化アルゴリズムが符号化されて格納される。暗号化鍵値６３－３のフィールドには、マイクロプロセッサ２０が暗号化プログラムを解読するために使う共通鍵Ｋｘが、マイクロプロセッサ２０に固有の公開鍵２１４で暗号化された値ＥＫｐ［Ｋｘ］が格納されている。共通鍵Ｋｘの長さは、たとえば、暗号にトリプルＤＥＳを使った場合には１９２ビットが必要になる。そして、ＥＫｐ［Ｋｘ］は、マイクロプロセッサ２０の公開鍵２１４によって暗号化される際のブロック長の整数倍のサイズとなる。

【００６７】.textセクション６６は、共通鍵Ｋｘを用い、暗号化アルゴリズム６３－２のフィールドで指定されたアルゴリズムによって暗号化されている。ここでは、暗号化アルゴリズムには共通鍵方式のトリプルＤＥＳが指定されているものとする。特性フィールド６３－１には、プログラムが暗号化されていることを示すため、暗号化フラグが１にセットされていることは第１実施形態と同様である。

長いビット長を要する鍵情報を、直接ページテーブルのフィールドに割り付けると、ページテーブルの大きさが増大してメモリ２０３の利用効率が低下する。たとえば、ページが４Ｋバイトのときに、キャッシュラインサイズと同一の３２バイトの鍵をページテーブルの各ページに割り当てると、ページサイズの１％弱のメモリがページテーブルのために必要となってしまう。そこで、ページテーブルの大きさを増大させたとしても、命令ＴＬＢ２２０のサイズが以前と同じであれば、命令参照に対する命令ＴＬＢ２２０のヒット数が低下し、プログラムの実行速度が低下する。命令ＴＬＢ２２０のサイズも同時に大きくすることも考えられるが、高速な論理ー物理アドレス変換が要求される命令ＴＬＢ２２０の性質上、コスト高になってしまう。

【００７８】また、実際には、暗号化鍵の種類は必ずしもページごとに変更する必要はなく、プログラムの秘密保護のためには、たかだか数種類の鍵が使い分けられれば十分であることが多い。そこで、第２実施形態では、複数のページが同一の復号化鍵情報を共有できるように独立のキーテーブル７９を設け、ページテーブル７７にはキーテーブル７９への番号を格納することにより、ページテーブルル７７に必要なメモリ量を低減させたのである。

【００７９】さて、プログラムの配置とページテーブルの鍵情報の設定が終わると、ローダは.textセクションヘッダ６３が配置されたページテーブルの暗号化フラグを１にセットする。そのページテーブルが命令ＴＬＢ２２０にキャッシュされている場合にページテーブルの暗号化フラグが１にセットされると、バスインターフェイスユニット２２２がそれを監視していて、一次命令キャッシュ２１８および二次命令キャッシュ２０５の対応するエントリを無効化する。

【００８０】上述のキーテーブル７９の値は復号化鍵ＴＬＢ２６にも格納される。キーテーブル７９が書きかえられた場合は、対応する復号化鍵ＴＬＢ２３６のエントリも書き換えられる。ここで、復号化鍵ＴＬＢ２２０には、メモリ上の復号化鍵情報Ｅ［Ｋｘ］が、マイクロプロセッサ２０に固有の公開鍵２１４で復号化されて格納される。このとき、メモリ２０３上の復号化鍵情報Ｅ［Ｋｘ］の長さを、復号化鍵ＴＬＢ２２０上の復号化された鍵情報Ｋｘより長く取ることにより、鍵の安全性とＴＬＢ２２０の利用効率を両立させることができる。マイクロプロセッサ２０内部の復号化鍵ＴＬＢ２２０に格納された鍵情報は、ユーザプログラムからは見ることができないからである。もちろん、ユーザプログラムによるキーテーブルの明示的なメモリ参照に対しては、復号化された値Ｋｘではなく、もとの暗号化されたままの値Ｅ［Ｋｘ］が返される。

【００８１】キーテーブル７９の設定が完了すると、第１実施形態と同様に、ローダはプログラムのエントリポイントに制御を移し、プログラムの実行が開始される。プロセッサが命令として暗号化領域にアクセスすると、キャッシュのミスヒットが生じ、メインメモリ２０３から一次命令キャッシュ２１８への読み込みが始まる。このとき、ページテーブル７７からキーテーブル７９への参照で指定されたキーテーブル７９のエントリが、マイクロプロセッサ２０の秘密鍵２１２で復号化されて、復号化ＴＬＢ２３６に読み込まれる。復号化された鍵情報Ｋｘを使って、メインメモリ２０３に格納されていた暗号化データが復号化処理部２１６によって復号化され、一次命令キャッシュ２１８へと読み込まれ、実行される。

【００８２】以後に行われる実行終了後のキャッシュの無効化、割り込み処理などは第１実施形態と同様である。

【００８３】このように、第２実施形態ではプログラムの復号化に共通鍵方式を採用することができる。共通鍵のアルゴリズムは一般に同一鍵長さの公開鍵アルゴリズムと比較してハードウエア化が容易であり、復号化機能のコストを低減することができる。さらに、共通鍵をマイクロプロセッサに固有の公開鍵で暗号化し、その復号はマイクロプロセッサ２０に固有の秘密鍵でないとできないので、安全性も確保される。

【００８４】＜第３実施形態＞図９は、本発明の第３実施形態にかかるプログラム配布システム９０の図である。このプログラム配布システムは、基本的に、第１または第２の実施形態で説明したマイクロプロセッサを内蔵したコンピュータシステムにプログラムを配布するように設計されている。したがって、配布される実行プログラムは、マイクロプロセッサ固有の鍵で直接暗号化されるか、あるいは任意の共通鍵で暗号化され、共通鍵そのものををマイクロプロセッサ固有の鍵で暗号化して実行プログラムに添付する。このため、同一のプログラム媒体のコピー、たとえば大量にプレスされるＣＤ－ＲＯＭなどによってプログラムを配布することができない。そこで、ネットワークを介してプログラムを配布する。

【００８５】図９に示すプログラム配布システム９０は、ネットワーク９５と、ネットワーク９５に接続され、実行プログラムをネットワーク９５を介して配布するプログラム配布装置９３と、同じくネットワーク９５に接続され、ネットワークを介してプログラム配布装置９３から実行プログラムの配布を受けるクライアント装置９１とを含む。

【００８６】プログラム配置装置９３は、クライアント装置９１との間に第１の通信路を設定する第１通信路設定部９３１と、記第１の通信路を介してクライアント装置９１を使用するユーザのユーザ認証を行うユーザ認証部９３３と、第１の通信路上に、クライアント装置が内蔵するマイクロプロセッサに直接連絡する第２の通信路を設定する第２通信路設定部９３４と、第２の通信路を

【００９６】図１０のシーケンスにおいて、まずシーケンス１００１において、クライアント装置９１とサーバ９３との間に、それぞれの装置の第１通信路設定部により、安全な第１の通信路を設定する。具体的には、クライアント装置９１から、ネットワーク９５を介してサーバ９３に通信開始要求を送り、通信路の秘密を守るための鍵共有を行う。これは周知の鍵共有プロトコル、ＤＨ方式などでよい。以後のクライアント装置９１とサーバ９３との間の通信は、こうしてネットワーク９５に設定された、盗聴に対して安全な通信路を通じて行う。

【００９７】第１通信路が設定されたら、シーケンス１００２で、クライアント装置９１はサーバ９３に対してダウンロードしたいファイル（プログラム）を要求し、サーバ９３はクライアント装置９１とユーザレベルの認証や課金処理などを行う。ダウンロードの過程で、処理の秘密をユーザから守るため、以下に説明するクライアント装置９１でのダウンロードシーケンスの少なくとも一部は、暗号化コードとして実行される。ここでは、暗号化コードによって実行される部分を、マイクロプロセッサ９０１の動作として表現する。

【００９８】シーケンス１００３では、第１通信路上に、クライアント装置９１のマイクロプロセッサ９０１とサーバ９３とを直接接続する、安全な第２の通信路を設定する。

【００９９】本発明では、ユーザがダウンロード過程で処理されるデータの一部を不正に取得することを防止するため、ダウンロードプログラムでは、コード自身はもちろんのこと、処理過程でメモリ上に置かれるデータについても、ユーザが読み取り理解することが困難となるように記述されている。この目的をさらに強化するため、シーケンス１００３では、サーバ９３とマイクロプロセッサ９０１との間でも秘密鍵による通信の暗号化を行う。

【０１００】秘密鍵の共有を行わない場合、ユーザによる不正、たとえば、サーバ９３とマイクロプロセッサ９０１との間の通信メッセージで、マイクロプロセッサ９０１の持つ公開鍵Kpを偽の公開鍵にすりかえて暗号化されたプログラムを取得し、偽の公開鍵に対応する既知の秘密鍵で復号化することによって平文のプログラムがユーザの手にわたるおそれがあるからである。サーバ９３とマイクロプロセッサ９０１との間の通信を秘密鍵で暗号化することにより、上述のようなユーザの不正を防ぐことができる。以後、「マイクロプロセッサ９０１とサーバ９３との間の通信」という場合、マイクロプロセッサ９０１上の耐タンパなプログラムとサーバ９３とが共有する暗号鍵によって暗号化され、保護された通信を意味する。

【０１０１】安全な第２通信路の設定後、マイクロプロセッサ９０１とサーバ９３は相互認証を行う。すなわち、シーケンス１００４で、マイクロプロセッサ９０１はサーバ９３に対するチャレンジのための乱数Ｒｃを生成し、マイクロプロセッサ９０１に固有の公開鍵Kpとともに、第２通信路を介してサーバ９３に送信する。チャレンジを受け取ったサーバ９３は、シーケンス１００５で、サーバ９３の秘密鍵K'sで乱数Ｒｃを暗号化した署名$S_{K's}$［Ｒｃ］を、サーバ９３からのチャレンジＲｓおよびサーバ側公開鍵K'pとともに、マイクロプロセッサ９０１に送信する。（図１０では、署名$S_{K's}$［Ｒｃ］をＳ［Ｒｃ］(K's)と表記し、類似の表現も同様の表記とする。）

マイクロプロセッサ９０１は、サーバ９３から送られてきた署名$S_{K's}$［Ｒｃ］が、サーバの公開鍵K'ｐでハッシュ化された$V_{K'p}$［Ｒｃ］と一致するかどうかを確認する。不一致の場合は、サーバ９３の認証に失敗して、以後の処理を中止する。認証に成功すれば、シーケンス１００６でサーバ９３のチャレンジに対してS_{Ks}［Ｒｓ］を計算し、証明書Certをサーバ９３に送る。サーバ９３は、レスポンスS_{Ks}［Ｒｓ］をV_{Kp}［Ｒｓ］と比較して、一致しなければ処理を中止する。一致した場合は、マイクロプロセッサ９０１の公開鍵Ｋｐと認証機関の公開鍵Kvalとから、V_{Kval}［Ｋｐ］を計算し、証明書Certから公開鍵に対する署名S_{Kcert}［Ｋｐ］と照合し、検証に失敗すれば処理を中止する。

【０１０２】照合できた場合は、$E_{K'p}$［Cert］を復号化してCertを取り出し、検証する。ここでも検証に失敗すれば処理を中止する。検証に成功して、証明書によりマイクロプロセッサ９０１が公開鍵Ｋｐを持つことを確認すると、サーバ９３は実行プログラムをマイクロプロセッサ９０１の公開鍵Ｋｐで暗号化した暗号化プログラムE_{Kp}［Ｐｒｏｇ］を作成する。このとき、第１実施形態で説明したように、プログラムのコード部分をマイクロプロセッサ９０１の公開鍵Ｋｐで暗号化する。このとき、第１実施形態で説明したように、プログラムのコード部分をマイクロプロセッサ９０１の公開鍵Ｋｐで暗号化する。暗号化にあたっては、第１実施形態で説明したように、プログラム本体の.textセクションは暗号化するが、ジャンプテーブルの.IATセクションは平文のままとしておく。

【０１０３】シーケンス１００７で、サーバ９３は暗号化プログラムE_{Kp}［Ｐｒｏｇ］と、サーバ９３自身の秘密鍵K'sによる署名$S_{K's}$［E_{Kp}[Prog]］とを、第２通信路を介してマイクロプロセッサ９０１に送る。この暗号化プログラムと署名は、直接マイクロプロセッサ９０１とサーバ９１との間に確立された第２通信路を介して送られるため、クライアント装置９１はこれを傍受することはできない。

【０１０４】マイクロプロセッサ９０１はプログラムの受信を完了すると、シーケンス１００８でクライアント装置９１にダウンロード終了を通知する。シーケンス１

プロセッサ９０１に送信する。このとき、第２通信路設定後に、マイクロプロセッサ９０１に備えられている暗号復号化能力をサーバ側からマイクロプロセッサ９０１に問い合わせるシーケンスを追加して、マイクロプロセッサ９０１に処理可能な暗号化アルゴリズムの中からサーバ９３が暗号化アルゴリズムを選択してもよい。

【０１１３】実行コードの復号化にプログラムごとに使い捨ての共通鍵を使うので、鍵の長さを短くし、クライアント装置９１のメモリあるいはマイクロプロセッサ９０１のキャッシュの中に構築されるページテーブルのサイズを小さくできる。

【０１１４】上述したようなダウンロードの手順は、データベースや顧客情報などの秘密情報の取り扱いにも適用できる。また、プログラムをターゲットのマイクロプロセッサだけで実行できるように暗号化することを除けば、実行プログラム以外の任意の音楽、映像データにも拡張して適用可能である。

【０１１５】

【発明の効果】本発明のマイクロプロセッサによれば、マイクロプロセッサが組み込まれたＰＣの所有者に解析されることなく、暗号化されたプログラムを安全に実行することができる。また、既存のプログラムにソース変更や再コンパイルすることなく、完全にプロセッサ内部で復号化を行うことができる。

【０１１６】本発明の別の形態のマイクロプロセッサによれば、プログラムの暗号化に共通鍵を用い、共通鍵自体をマイクロプロセッサの公開鍵で暗号化した。これにより暗号化プログラムのセキュリティが補償されるとともに、マイクロプロセッサのハードウエアサイズを大幅に縮小することができる。

【０１１７】本発明のプログラム配布システムでは、ネットワークを介して、プログラム配布装置からクライアント装置へ、安全かつ確実に暗号化プログラムを配布することができる。また、マイクロプロセッサで実行される耐タンパなダウンロードプログラムの存在を前提とすることにより、第三者を介さず、直接プログラム配布装置とマイクロプロセッサとの間で、プログラムのダウンロードを安全かつ効率良く行うことができる。

【０１１８】また、コンピュータ読み取り可能な記録媒体に、プログラム本体部分を暗号化して記録し、外部のプログラムへの直接の参照を行うＩＡＴ領域は暗号化せずに平文の状態で格納することにより、プログラム実行時のリロケーションが正しく行われることを可能にする。

【図面の簡単な説明】

【図１】本発明の第１実施形態に係るマイクロプロセッサの概略ブロック図である。

【図２】本発明の第１実施形態に係るマイクロプロセッサで実行される暗号化プログラムのファイル形式の一例を示す図である。

【図３】本発明の第１実施形態に係るマイクロプロセッサの物理アドレスと、仮想記憶を管理するページテーブルと、論理アドレスとの対応関係を示す図である。

【図４】図３に示すページテーブルの中の、ページエントリの一例を示す図である。

【図５】本発明の第２実施形態に係るマイクロプロセッサの概略ブロック図である。

【図６】本発明の第２実施形態に係るマイクロプロセッサで実行される暗号化プログラムのファイル形式の一例を示す図である。

【図７】本発明の第２実施形態に係るマイクロプロセッサの物理アドレスと、仮想記憶を管理するページテーブルと、キーテーブルとの対応関係を示す図である。

【図８】図７に示すページテーブルとキーテーブルとの関係を詳細に示す図である。

【図９】本発明の第３実施形態に係る、プログラム配布システムの図である。

【図１０】本発明の第３実施形態に係るプログラム配布のシーケンス図である。

【図１１】図９に示すクライアント装置のプログラム受け取り手順を示すフローチャートである。

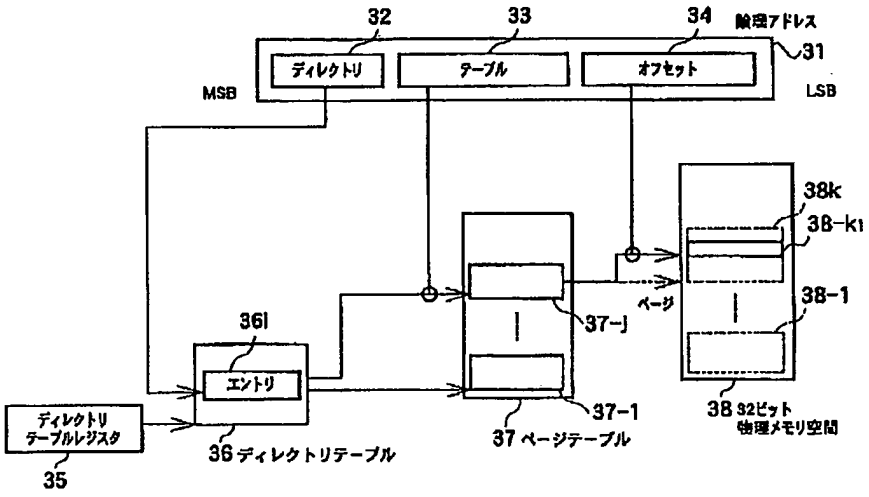
【図１２】図９に示すプログラム配布装置のプログラム配布手順を示すフローチャートである。

【図１３】図９に示すクライアント装置内部のマイクロプロセッサの動作を示すフローチャートである。

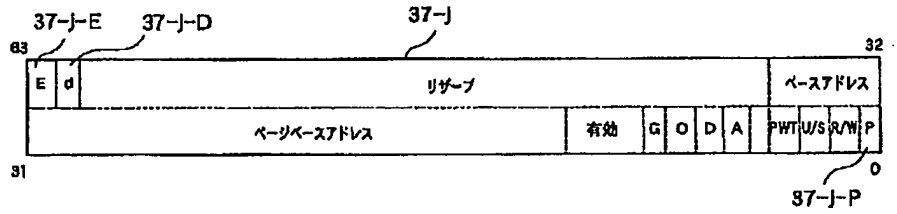
【符号の説明】

１０、２０、９０１　マイクロプロセッサ
３７、７７　ページテーブル
７９　キーテーブル
９１　クライアント装置
９３　プログラム配布装置
９５　ネットワーク
１０３、２０３　メインメモリ
１０５、２０５　二次キャッシュ
１１２、２１２　秘密鍵
１１４、２１４　公開鍵
１１６、２１６　復号化処理部
１１８、２１８　一次命令キャッシュ（第２の記憶手段）
１２０、２２０　命令ＴＬＢ（第１の記憶手段）
１２２、２２２　バスインターフェイスユニット
１２４、２２４　レジスタ
１２６、２２６　レジスタ値暗号化／復号化処理部
２３６　復号化鍵ＴＬＢ
９０５、９３２　第１通信設定部
９０６、９３４　第２通信設定部
９１０、９３３　ユーザ認証部
９０８、９１４　プログラム受信部
９３５　プログラム認証部
９３６　プログラム暗号化処理部

【図３】

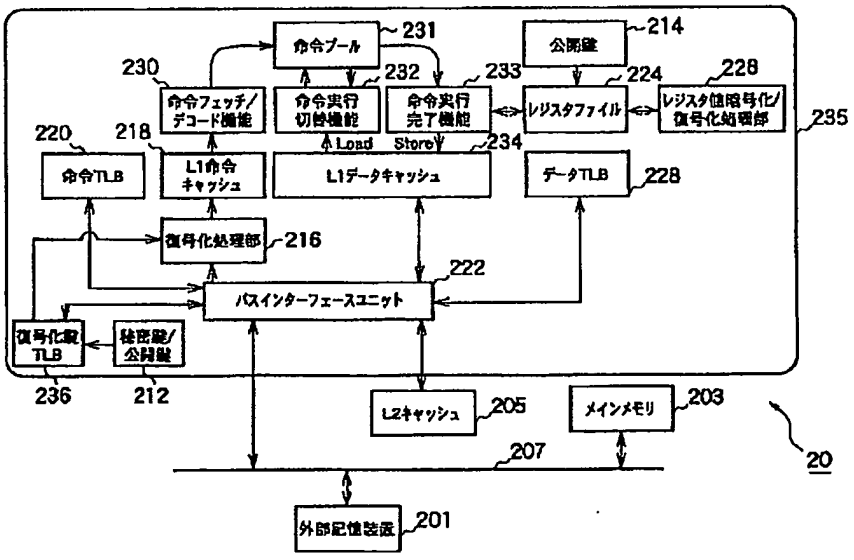


【図４】

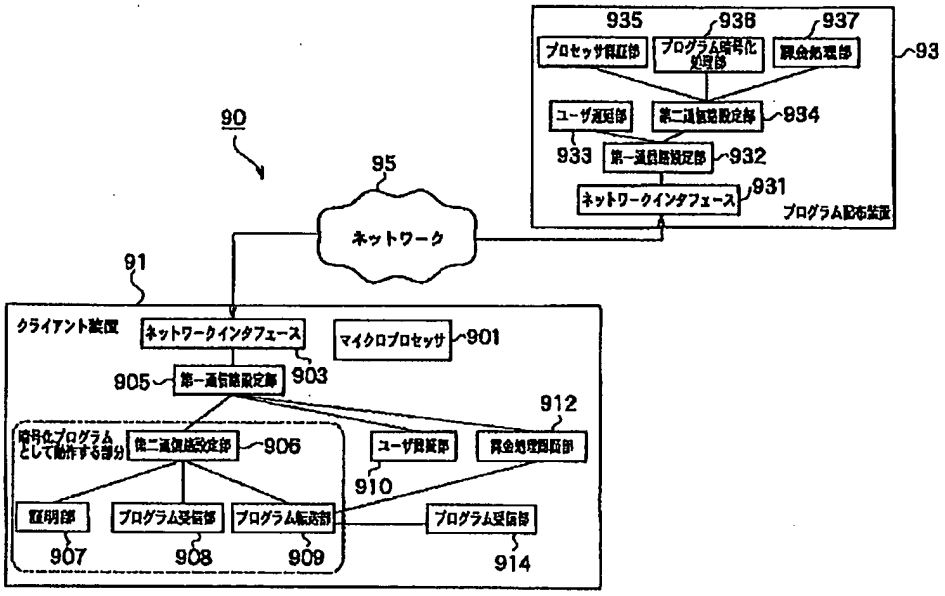


ページテーブルエントリ

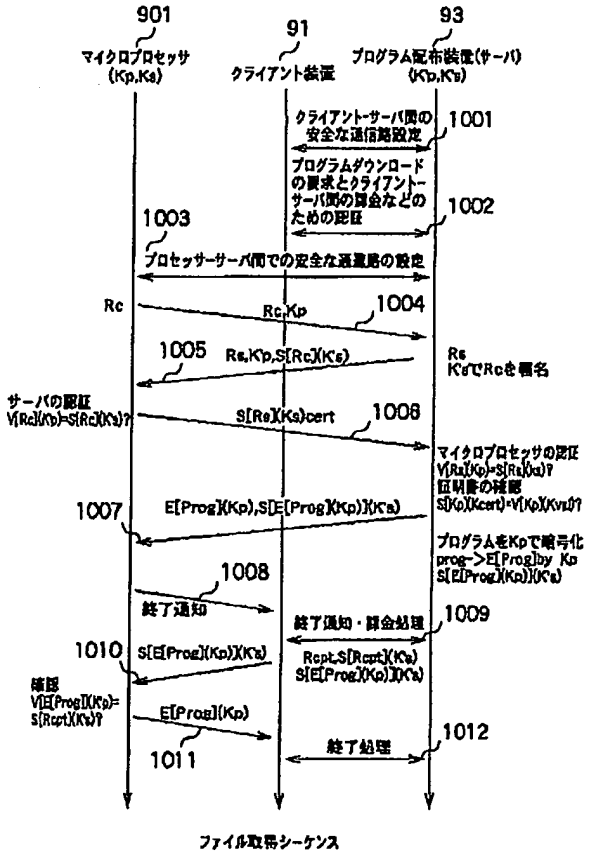
【図５】



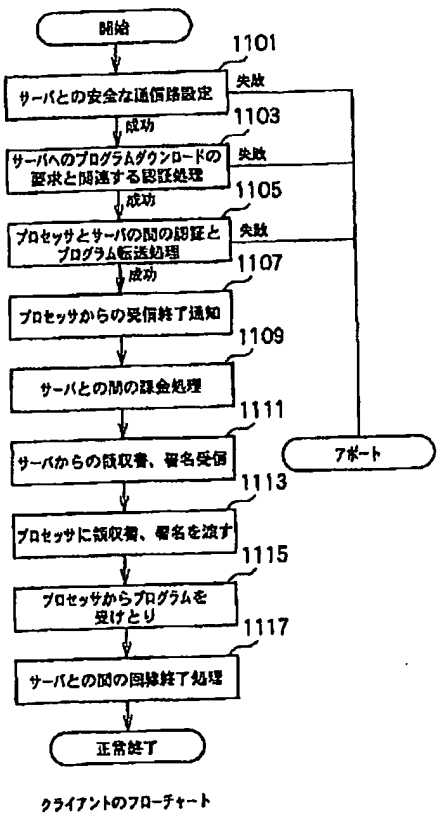
【図９】


935
936
937
プロセッサ認証部
プログラム暗号化処理部
課金処理部
93
ユーザ認証部
第二通信路設定部
934
933
第一通信路設定部
932
ネットワークインタフェース
931
プログラム配布装置
90
95
ネットワーク
91
クライアント装置
ネットワークインタフェース
マイクロプロセッサ
901
905
第一通信路設定部
903
912
暗号化プログラムとして動作する部分
第二通信路設定部
906
ユーザ認証部
課金処理開始部
910
証明書
プログラム受信部
プログラム転送部
プログラム受信部
907
908
909
914

【図１０】


901
マイクロプロセッサ (Kp,Ka)
91
クライアント装置
93
プログラム配布装置(サーバ) (Kp,Ks)
クライアントーサーバ間の安全な通信路設定
1001
プログラムダウンロードの要求とクライアントーサーバ間の課金などのための認証
1002
1003
プロセッサーサーバ間での安全な通信路の設定
Rc
Rc,Kp
1004
1005
Rs,Kp,S[Rc](Ks)
Rs Ksでを署名
サーバの認証 V[Rs](Kp)=S[Rs](Ks)?
S[Rs](Ks)cert
1006
マイクロプロセッサの認証 V[Rs](Kp)=S[Rs](Ks)? 証明書の検証 S[Kp](Kcert)=V[Kp](Kvs)?
1007
E[Prog](Kp),S[E[Prog](Kp)](Ks)
プログラムをKpで暗号化 prog->E[Prog]by Kp S[E[Prog](Kp)](Ks)
1008
終了通知
終了通知・課金処理
1009
1010
S[E[Prog](Kp)](Ks)
Rcpt,S[Rcpt](Ks) S[E[Prog](Kp)](Ks)
検証 V[E[Prog](Kp)= S[Rcpt](Ks)?
E[Prog](Kp)
1011
終了処理
1012
ファイル取得シーケンス

【図１１】


開始
1101
サーバとの安全な通信路設定
失敗
成功
1103
サーバへのプログラムダウンロードの要求と関連する認証処理
失敗
成功
1105
プロセッサとサーバの間の認証とプログラム転送処理
失敗
成功
1107
プロセッサからの受信終了通知
1109
サーバとの間の課金処理
1111
サーバからの領収書、署名受信
アボート
1113
プロセッサに領収書、署名を渡す
1115
プロセッサからプログラムを受けとり
1117
サーバとの間の回線終了処理
正常終了
クライアントのフローチャート